Aufgabenblatt 4

Das sind die Aufgaben der Woche 4. Es handelt sich um Loops (Schleifen) in Programmen, also for und while.

# Aufgabe 16 - Schleifen

Implementieren Sie Python-Code für eine Schleife, die für jede 3er-Potenz die letzten 3 Ziffern der zugehörigen Dezimalzahl ausgibt. Die Anzahl der Schleifendurchläufe soll hierbei durch eine Nutzereingabe festgelegt werden. **(2 P.)**Was ist die kleinste Dreierpotenz grösser 3, die auf „001“ endet? **(1 P.)**Können Sie das gleiche Problem auch für 7er-Potenzen lösen? **(1 P.)**

Lösung:

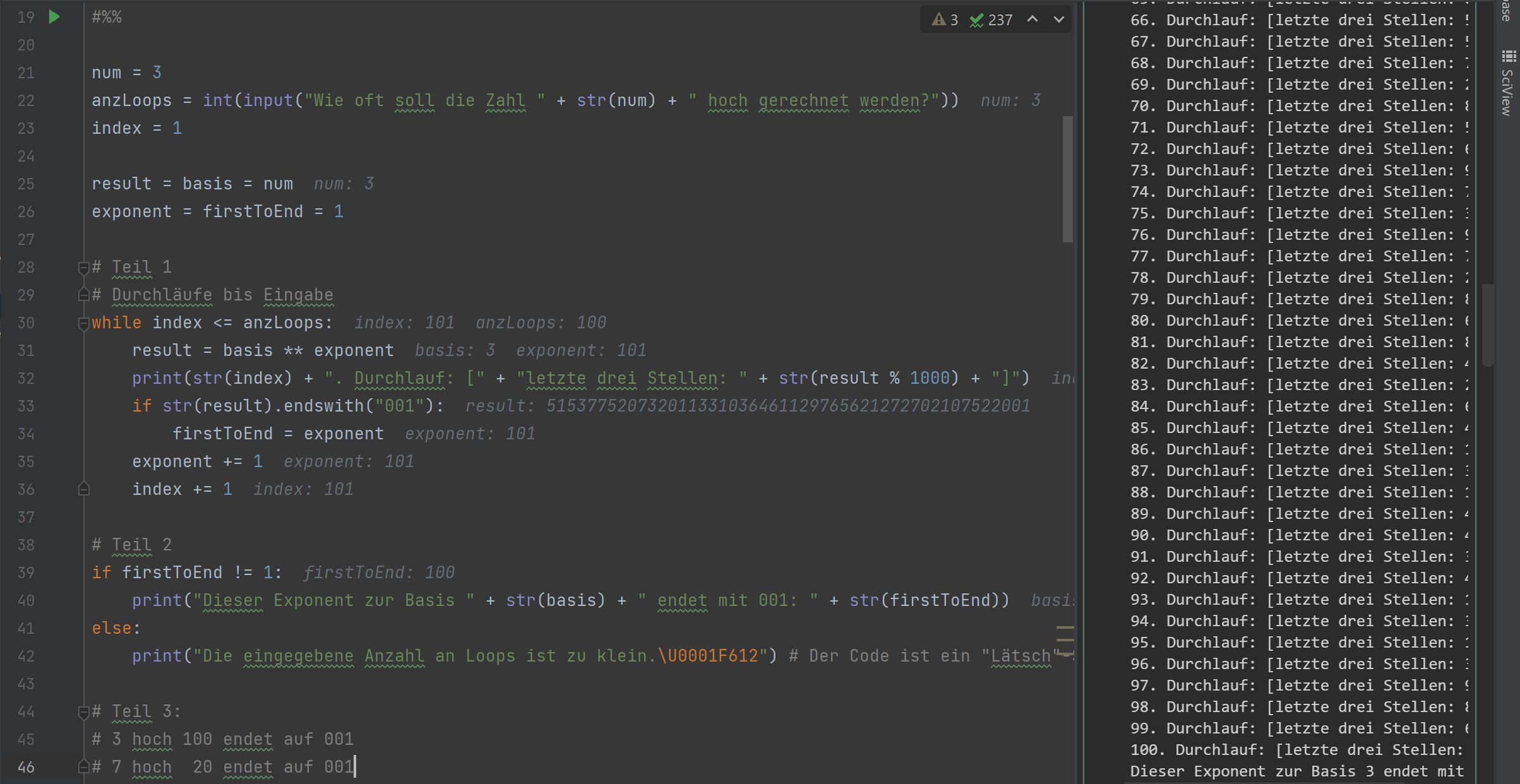
Die Lösung ist in diesem Screenshot zu sehen, wie auch im Jupyter Notebook oder im Aufgabe16.py File (befindet sich im .zip) zu finden.  


Abbildung 1 - Aufgabe 16

Wichtig zu vermerken ist:

* Im Teil 2 am Ende wird, falls die Zahl zu klein ist, ein Emoticon ausgegeben.
* Im Aufgabe16.py File ist der Algorithmus in einer Funktion und am Ende 2-mal ausgegeben, einmal mit der Basis 3 und einmal mit der Basis 7
* Die Respektiven Lösungen zur Fragen 2 und 3 sind:
* Potenz von 3 welche auf 001 endet: 100
* Potenz von 7 welche auf 001 endet: 20

# Aufgabe 17 - Zufallsstring

Schreiben Sie Python-Code, der eine beliebige zufällige Zeichenkette einer nutzerdefinierten Länge generiert. Alle Zeichen in der Zeichenkette sollen aus einem nutzerdefinierten String sein, also beispielsweise Ihr Name. **(4 P.)**

Lösung:

Auf dem Screenshot ist die Lösung zu sehen:

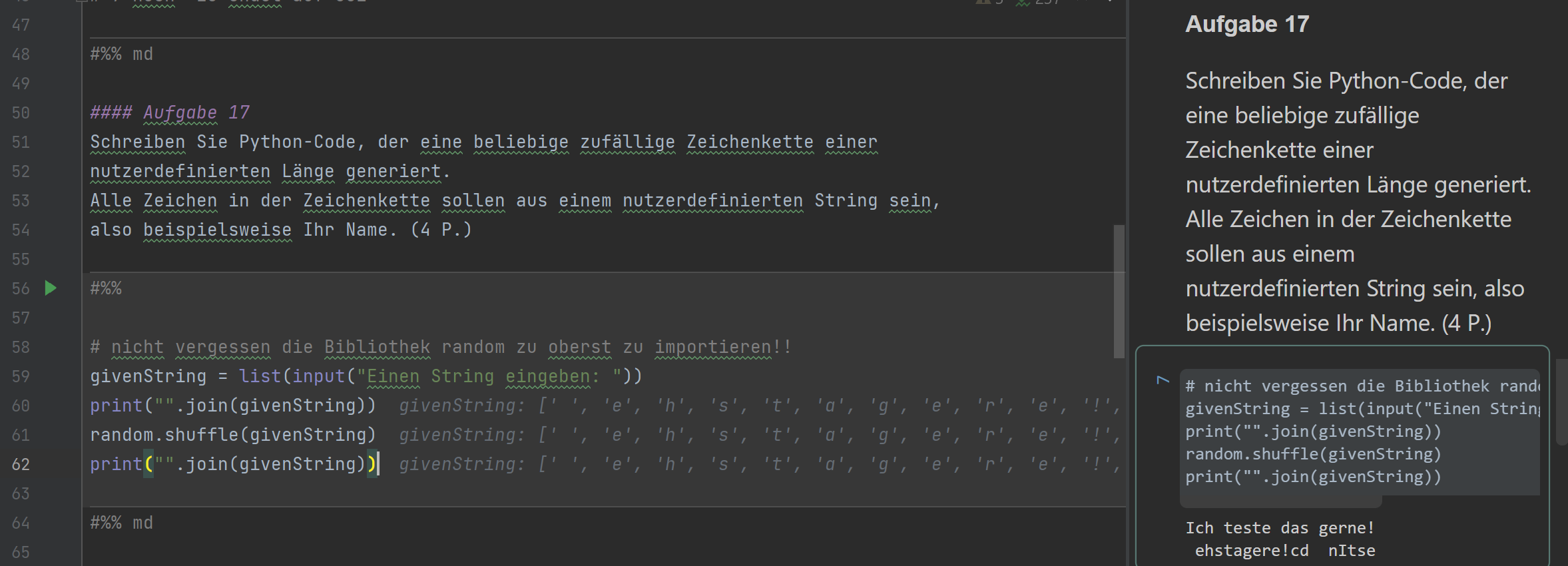


Abbildung 2 - Aufgabe 17

Zu beachten ist, dass am Anfang des Codes die Bibliothek random importiert werden muss und dass der eingegebene String als Liste gespeichert wird. Das ermöglicht das einfache shuffeln und joinen der einzelnen Charakter.

Es gibt noch weitere Bibliotheken, welche es ermöglichen eine Zeichenkette zu randomisieren bzw. zu shuffeln:

* Numpy
* String\_utils
* More\_itertools

Alle diese Bibliotheken besitzen eine shuffle Funktion.

# Aufgabe 18

Berechnen Sie alle Produkte von i\*j\*k für 0 < i <= 10, 0 < j <= 15, 0 < k <= 20, für die das Produkt i\*j\*k eine Quadratzahl ist. Geben Sie die Lösungsprodukte aus. **Hinweis zur Einfachheit der Aufgabe:** Eine ganze positive Zahl ist eine Quadratzahl, wenn ihre Quadratwurzel eine ganze Zahl ist. **Alternativ:** Eine ganze positive Zahl ist eine Quadratzahl, wenn sie als Summe von aufeinanderfolgenden ungeraden Zahlen, beginnend bei 1 geschrieben werden kann (Beispiel: 1 + 3 + 5 + 7 + 9 = 25). **(4 P.)**

Lösung:

Meine Lösung funktioniert folgenderweise:

1. Alle drei Variabeln in eine for-Schleife setzen
2. Das resultat der drei Variabeln berechnen (Zeile 88)
3. Das Resultat evaluieren
   1. Wurzel bilden
   2. Wurzel auf Ganzzahligkeit überprüfen
4. Ergebnis ausgeben

So werden ALLE Produkte berechnet und nur die Quadratischen ausgegeben. (siehe unten)

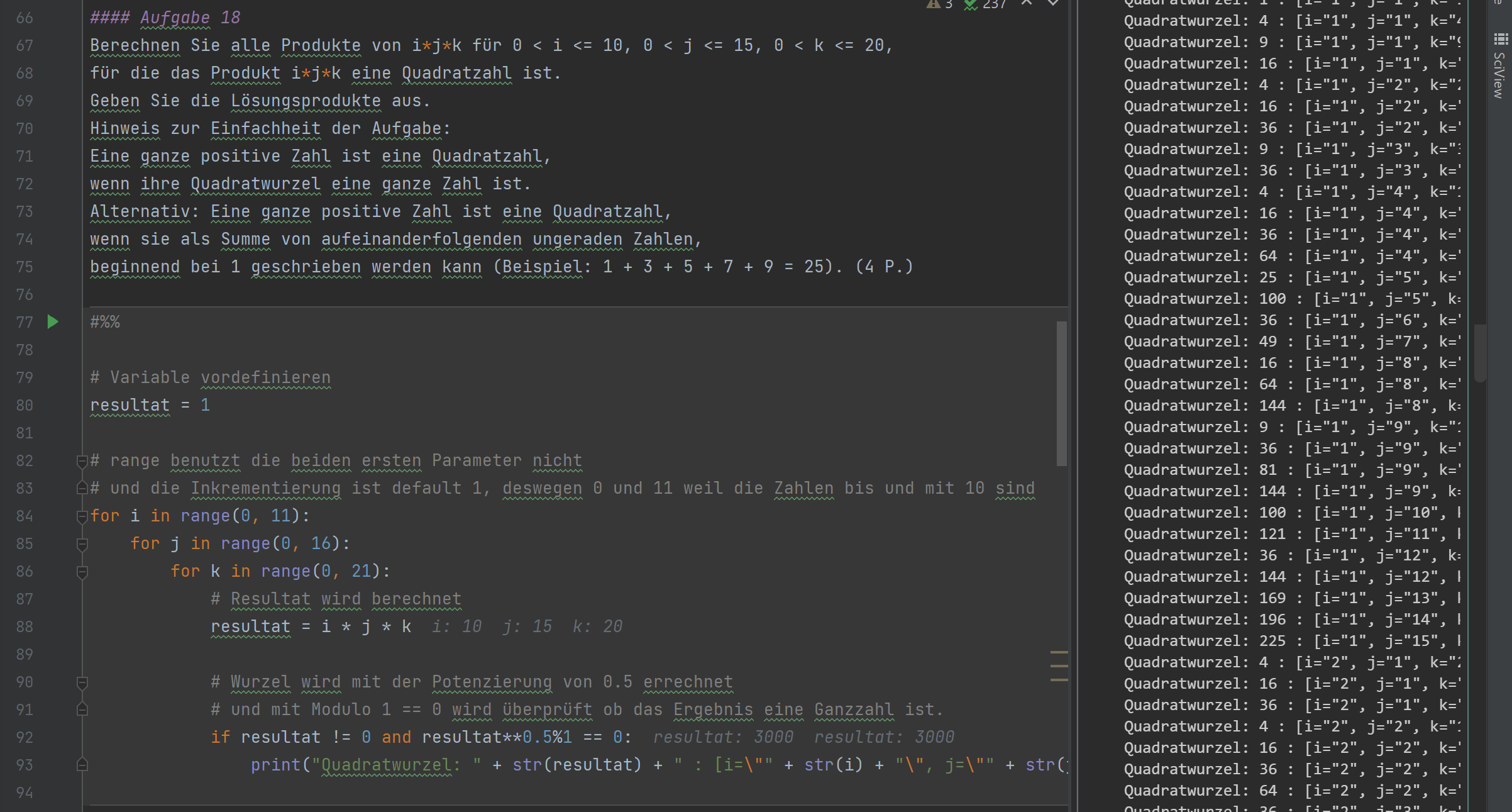


Abbildung 3 - Aufgabe 18

Wichtig zu erwähnen ist, dass das Ergebnis auf der rechten Seite um einiges Grösser ist als in der Abbildung 3 zu sehen.

Wenn das alle Quadratzahlen gesehen werden wollen, so kann das Python File Aufgabe18.py dazu verwendet werden.

# Aufgabe 19

Wandeln Sie folgende for-Schleife in eine while-Scheife um, die die gleiche Ausgabe wie die for-Schleife erzeugt. **(4 P.)**



Abbildung 4 - Aufgabenstellung Aufgabe 19

Lösung:

Die Lösung ist eine Umwandlung der Zeile in welcher das for verwendet wird. Grundsätzlich sind folgende Dinge zu beachten:

1. Range beginnt nicht bei 0
2. Die Variable j wird um i erhöht, i wird bei jeder Iteration ebenfalls erhöht
3. Im Print werden die einzelnen j mit einem Abstand verkettet

Die Lösung mit der while-Schleife kann nun beispielsweise folgendermassen aussehen:

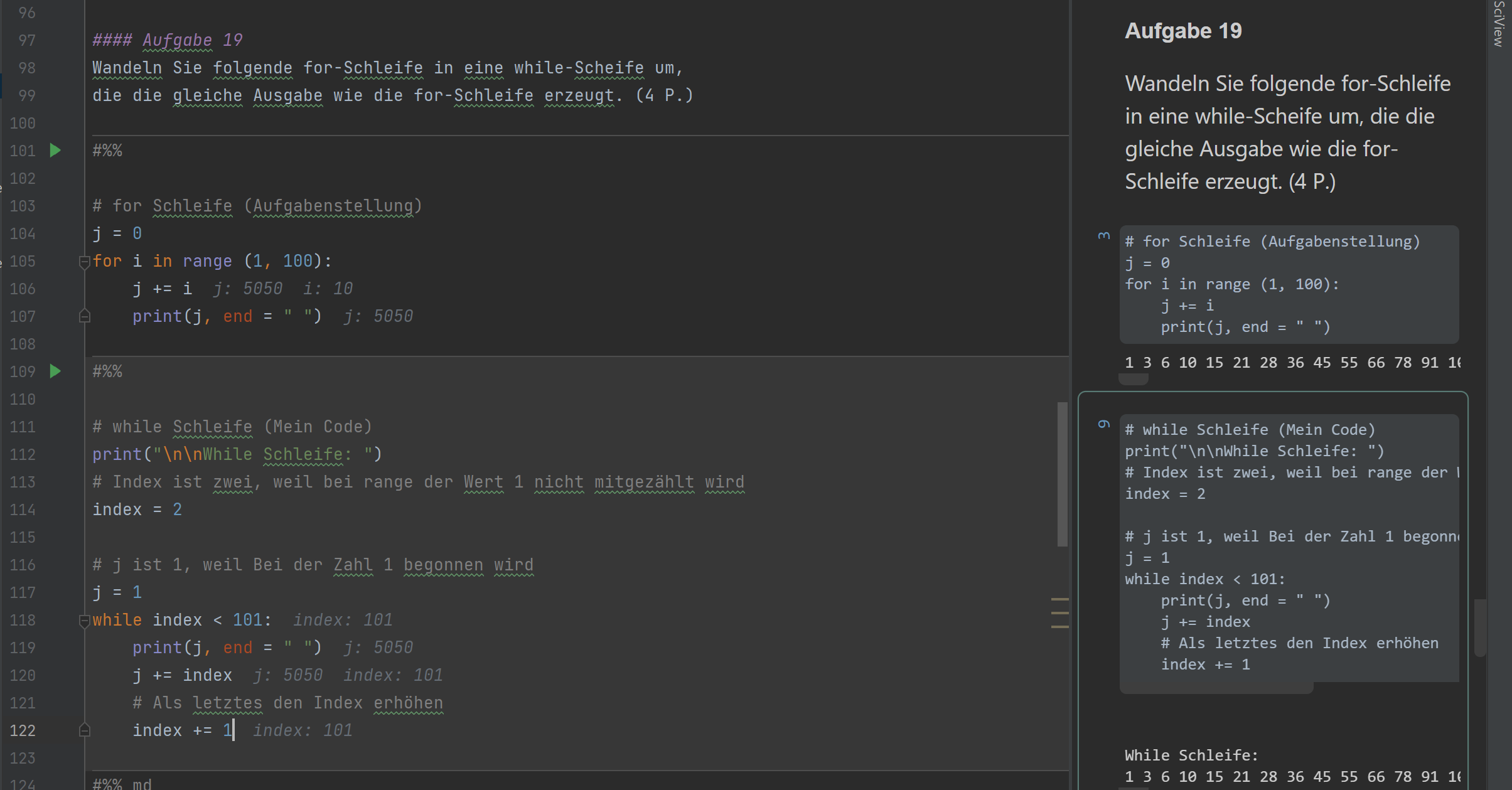


Abbildung 5 - Aufgabe 19

Alternativ kann das ganze auch als Rekursion gelöst werden:

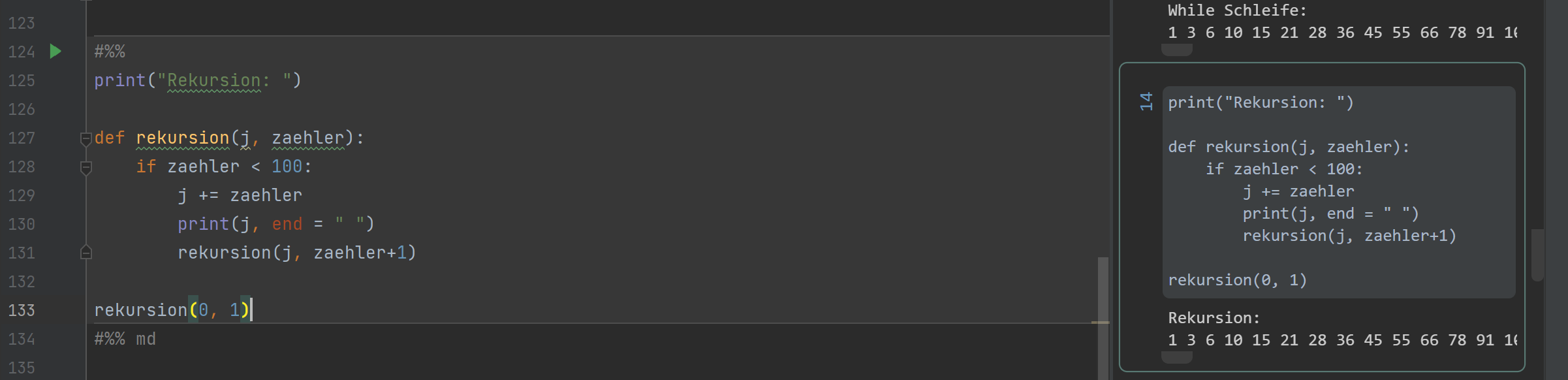


Abbildung 6 - Aufgabe 19 als Rekursion

Das ist nicht nach Aufgabenstellung, aber trotzdem schön.

# Aufgabe 20

Wie oft muss man einen Würfel im Durchschnitt würfeln **(random)**, bis zum ersten Mal 20 mal eine 6 gewürfelt wurde?   
Schreiben Sie Python-Code, der ein solches kleines Häufigkeitsexperiment durchführt.   
Wie viele Teilexperimente müssen Sie durchführen? Diskutieren Sie. (Beispiel, siehe Vorlesung). **(4 P.)**

Lösung:

Mein Code wählt zufällige Zahlen von 1 bis 6 und speichert den jeweiligen Wert in einer Variablen. Dieser Wert (in der Variablen) wird überprüft. Falls er eine 6 ist, dann wird die Variable anzSechser um 1 erhöht.   
Zum Schluss wird bei jedem Durchgang die Variable anzWurf um 1 höher gezählt, damit festgehalten wird, wie oft gewürfelt werden musste bis 20 6er gewürfelt wurden:

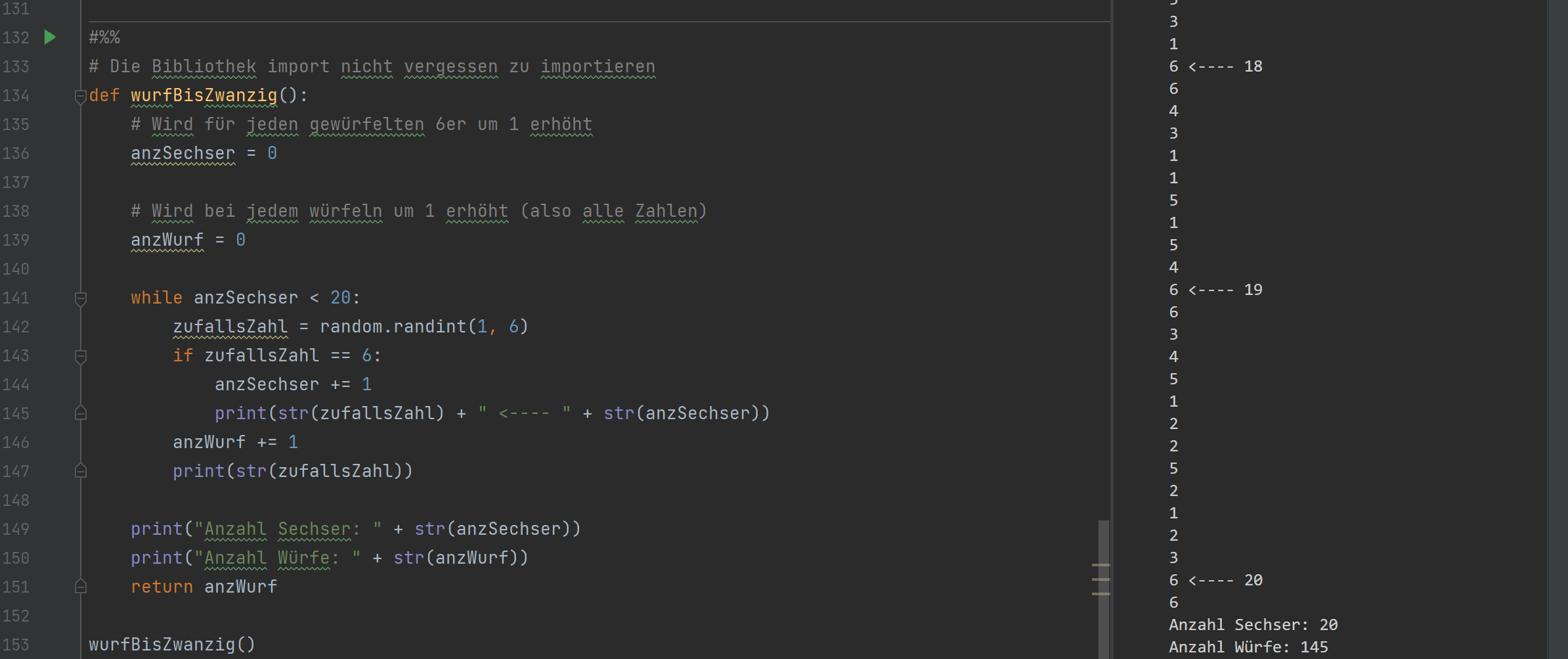


Abbildung 7 - Aufgabe 20

Die While Schleife ist zu Ende, wenn 20-mal ein 6er gewürfelt wurde.

In der Zip Datei ist nebst der Python Datei Aufgabe20.py noch die Python Datei Aufgabe20\_Haufigkeit zu finden. In dieser Methode wird die oben gezeigt Funktion mehrmals durchprobiert und die Anzahl an benötigten Würfel-Versuchen addiert und durch die Anzahl an Durchläufen geteilt. So wird ein Mittelwert errechnet, welcher die Ungefähre Anzahl an Würfel-Versuchen anzeigt, welche benötigt werden, um 20-mal eine 6 zu würfeln.

Dieser Wert ist etwa 120. Der Grund dafür ist einfach mathematisch zu beweisen:

Allgemeine Chance einer 6: 1/6

20-mal eine 6: 1/6 / 20 🡪 1/120

**ACHTUNG**: Wenn die Anzahl der Durchgänge zu hoch ist, kann das Programm durchaus 5 Minuten benötigen, um die Anzahl zu berechnen. (Weil jeweils noch Ausgaben getätigt werden)